

**SYSTEM FOR CUSTOMIZING ON-LINE
COMPUTER SERVICES**

Docket Number: 10006790-1

Inventors:

Svend Frolund
Citizenship: Denmark
533 Victory Avenue
Mountain View, CA 94043

and

James Christopher Pruyne
Citizenship: U.S.A.
2424 Lexington Lane
Naperville, IL 60540

Contact: Thomas X. Li
(650) 857-5972

SYSTEM FOR CUSTOMIZING ON-LINE COMPUTER SERVICES

TECHNICAL FIELD

The present invention relates generally to computer systems and more particularly to non-functional properties of computer systems.

BACKGROUND ART

The operation of a computer system can be described in many ways. The most obvious ways to describe a system is to describe its applications or the functions it performs. That is, what the computer system actually does, such as manage bank accounts, airline reservations, and so on. The Quality of Service (QOS) or so-called "non-functional" aspects of a computer system can also be described. The term non-functional refers to all aspects of the system other than the basic function for which the system was built. Examples of non-functional properties include performance (i.e. how quickly the system performs its function), reliability (i.e. how often the system successfully performs its function), security, and so forth. Because the non-functional aspects of a system are somewhat less obvious, they also often receive less attention by designers of computer systems.

One approach to improving the non-functional characteristics of a computer system is to mingle the implementation of the desired non-functional properties with the functional properties. Commonly, programmers hardwire a single set of non-functional properties into a computer system. Moreover, the logic that implements the non-functional properties is rarely specified in one place, but is instead scattered as point solutions across the algorithms, which implement the purely functional aspects of the system. Frequently, when systems are built this way, the designers are not even aware of the distinction between functional and non-functional properties.

There are several disadvantages to the above approach. First, it is hard to change the non-functional properties of a system to adapt to varied requirements that arise in the same environment if they are implemented as a point-solution. Second, because programs typically contain point-solutions to non-functional issues, and because these point solutions are hard to identify in their own right, it is hard to alter the non-functional behavior of a system as it is deployed in varied environments. Third, it is hard to reuse the logic, which implements non-

functional properties, because a design that is well provisioned for a particular non-functional property as it relates to one application may not be at all suitable to another application. Therefore, the work on this particular non-functional property cannot benefit future applications. Likewise, later developments to improve non-functional properties cannot be easily applied to pre-existing applications.

Another approach has been to improve the characteristics of specific components of a system that applications depend upon. For example, the performance of an application that uses a database can be improved by improving the performance of the database system that the application uses. The typical difficulty with this approach is that by changing the underlying sub-system, the behavior for all applications that use that sub-system are changed. It is not possible to selectively choose which applications or under what circumstances these applications should use the modifications. In many cases, augmentations involve trade-offs so not having the flexibility to determine when to use them is a serious problem.

A further approach, often referred to as aspect-oriented programming, has been to provide an application developer with a way to specify the desired or required non-functional properties of their application. A development system is then used to merge the function of the application with these desired properties to generate the final application. However, as with the previous approach, it does not allow for new decisions to be made after the application is completed. Those non-functional properties specified by the developer are now part of the application, and cannot be modified later.

Also in the past, object-oriented middleware systems have been developed so that developers are able to build distributed applications almost as easily as non-distributed applications. This simplification has been the primary goal of middleware, and in particular, it has concentrated on the functional aspects of a distributed application and little effort has been made to support developers of non-functional aspects.

Middleware is, however, a good fit for supporting non-functional requirements because it has very nearly an end-to-end view of the system. Additionally, a great deal of semantic information such as application interface definitions and pairing of request and response messages is available. Because of this, many research efforts have focused on supporting non-functional requirements in middleware. These generally fall into two categories. The first category is to embed enhancements for specific properties into the middleware. The second category is to make non-functional concerns explicit in the development of an application. Both categories have similarities with aspect-oriented

programming. A specification is written in parallel to the application specification, and the middleware merges these to create an augmented application. Unfortunately, these still suffer from some of the disadvantages described above.

There seemed to be insurmountable problems with developing a middleware system with adequate support for non-functional properties because no assumptions about application specific information, such as interface definitions, are available to developers of non-functional components. Further, it is not possible to know *a priori* what non-functional properties will be required for particular deployments or instances of services, clients, or client-service pairs. The solution must be general enough to enable a variety of customizations without becoming so complex or specific that it is difficult to use and it must take into account possibly conflicting goals, such as high speed versus security. Further, there must be client/server neutrality for use on either the client or the server side of a distributed interaction and the solution must be independent of any particular middleware.

Those skilled in the art have long sought a solution, but the solution has long eluded them.

DISCLOSURE OF THE INVENTION

The present invention provides a computer system and software for customizing on-line computer services. An application is able to interact with a sub-system between which an interceptor system can be interposed. The interceptor system includes a proxy having an interaction with the application, a customization module providing a customization and having an interaction with the proxy and a dispatcher having an interaction with the customization module and having an interaction with the sub-system. A customization repository contains the customization modules and has interaction with a service and a client sharing the application and the sub-system. A control console has an interaction with the customization repository and the service for causing customization of the service and the client by disposing the proxy to interact with the application and the dispatcher interact with the sub-system. It is easy to change the non-functional properties of the system to adapt to varied requirements that arise in the same environment, to alter the non-functional behavior of the system as it is deployed in varied environments, and to reuse the logic, which implements non-functional properties. The invention is general enough to enable a variety of customizations without becoming so complex or specific that it is difficult to use and it takes into account possibly conflicting goals, such as high speed versus security. Further, there is

client/server neutrality for use on either the client or the server side of a distributed interaction and the software is independent of any particular middleware.

The above and additional advantages of the present invention will become apparent to those skilled in the art from a reading of the following detailed description when taken in
5 conjunction with the accompanying drawings.

BRIEF DESCRIPTION OF THE DRAWINGS

FIG. 1 (PRIOR ART) shows a block diagram of a prior art software system;

FIG. 2 shows a block diagram of a software system in accordance with the present invention; and

FIG. 3 shows a computer system in accordance with the present invention.

BEST MODE FOR CARRYING OUT THE INVENTION

Referring now to FIG. 1 (PRIOR ART), therein is shown a block diagram of a software system 8 having an interaction between an application 10 and a sub-system 12. The application 10 is a logical entity, which a human interacts with and which can be thought of as a computer application program. The sub-system 12 can be one of a number of sub-systems of different types, such as a database, a computer operating system, or a computer network.

The application 10 is able to "invoke" 14, as represented by a down-pointing arrow, the sub-system 12. The invoke 14 indicates that the application 10 will use the sub-system 12
20 to perform some function for which the sub-system 12 has been specifically designed. For example, a database that manages data for system storage such as a disc drive. bBy invoking the sub-system, the application is making a request to the sub-system to perform a specific function.

The functions that the sub-system 12 can perform are referred to as its "interface" 16,
25 as represented by a diamond-headed arrow. The interface 16 specifies the method by which the sub-systems may communicate and is essentially the list of the operations that the sub-system 12 can perform for the application 10.

This layered structure in FIG. 1 along with the well-defined interfaces provides an opportunity to transparently customize or alter the interactions between the application and
30 the sub-systems it utilizes.

Referring now to FIG. 2, therein is shown a block diagram of a software system 18 having an interceptor system 20 according to the present invention. The interceptor system 20 includes a proxy 22 having a first proxy interface 24, which substitutes the interactions of the proxy 22 for the interactions of the sub-system 12 with application 10. The computer language description of the interface 16 is used to generate the first proxy interface 24 to be the same as the interface 16. Also shown is a plurality of sub-systems 12-1 through 12-k with respective interfaces 16-1 through 16-k, which are possible with the present invention and which will be described later.

The proxy 22, under the control of a customization control 23, facilitates dynamic customization of the behavior of the application 10 for a plurality of customization modules 26-1 through 26-n. To do this, the proxy 22 performs two specific functions.

First, the proxy 22 prepares interactions between the application 10 and the sub-system 12 to be customized. This is done by collecting all of the data related to the interaction into a standard representation that can be understood by the plurality of customization modules 26-1 through 26-n, as will later be explained. This process is important because the standard representation allows customization logic to be developed independently of the application 10 or the sub-system 12. The customization will always work on the same representation of the interaction regardless of the application or sub-system in use.

Second, the proxy 22 provides for the management and control of the plurality of customization modules 26-1 through 26-n through the customization control 23. The proxy 22 provides a means of adding or removing the plurality of customization modules 26-1 through 26-n while the application 10 is running.

A management system 27 can install the plurality of customization modules 26-1 through 26-n for the application 10 at any time, even during run-time, by means of the customization control 23. This provides an immense degree of flexibility to the software system 18.

The plurality of customization modules 26-1 through 26-n have a plurality of first customization interfaces 28-1 through 28-n, which is common for interfacing with a second proxy interface 25 to allow for applications to be managed. For example, a customization module 26-2 that measures the current performance level as seen by the application 10 can be installed, and based on those performance measurements, performance customization modules 26-3 and 26-4 that are most suitable can be installed. To implement the control

interface, the proxy 22 maintains a set of active customization modules in a chain in which one customization module passes control on to the next.

Since the plurality of first customization interfaces 28-1 through 28-n are all the same, a single customization can to be used in any situation, regardless of the application or sub-system being used. Each of the plurality of customization modules 26-1 through 26-n is programmed to have some or all of the following capabilities as they inspect or alter the flow of invocations from the application 10 to the sub-system 12:

1. Error Handling – Each can intercept errors generated by the sub-system 12, and handle them before the errors are returned to the application 10.
2. Error Creation – Each may indicate a failure to the application 10, even if the underlying sub-system 12 has not generated or detected a failure.
3. Invocation inspection – Each may inspect the invocation to determine what data is being used during the invocation. This is possible because of the standardized representation of the invocation created by the proxy 22.
4. Inserting and extracting extra data or parameters – Each may insert or remove data into or out of the standard representation. It may do this to provide additional data to other the plurality of customization modules 26-1 through 26-n or to the sub-system 12 being invoked.
5. Short-circuiting / local handling – Each can prevent the invocation from reaching the sub-system 12 entirely if it should choose to do so and can send it to a different sub-system.

The customization modules 26-1 through 26-n have second customization interfaces (only second customization interface 29-1 shown) to interface with a first dispatcher interface 28 of a dispatcher 30. The first dispatcher interface 28 is the same as the first customization interfaces 28-1 to 28-n. The dispatcher 30 performs the inverse operation as the proxy 22. It receives an invocation from the plurality of customization modules 26-1 through 26-n, and transforms it into an invocation at a second dispatcher interface 32 for the interface 16 of the sub-system 12. This invocation may have been altered or customized by the customization modules 26-1 through 26-n preceding it in the interceptor system 20. In some cases, the dispatcher 30 may be used to further customize the invocation by directing it to a sub-system other than the one originally used by the application 10. An example of this might be forcing the invocation to cross a network rather than being performed on the local computer. One

use of this approach would be to access files stored on a remote computer instead of the files stored on a local computer and provided by the local file management sub-system.

In alternate embodiments, each of the plurality of first customization modules 26-1 through 26-n are developed to perform different non-functional operations and the customization control 23 provides different ways of managing the plurality of customization modules 26-1 through 26-n that are being used. For example, the customization control 23 can provide very specific control over matters such as the order in which the plurality of customization modules 26-1 through 26-n are invoked, their performance, failure modes, frequency of usage, system optimization, etc.

Referring now to FIG. 3, therein is shown a computer system 40 for adding customizations to services while the services are running in accordance with the present invention. The hub of the computer system 40 is a customization repository 42 where customizations are stored as executable (as opposed to source code only) software modules.

When a customization developer using a customization developer system 44 develops a new customization, it is uploaded along the link 46 to the customization repository 42, along with its description and additional meta-data such as whether it can be applied to clients, servers, or both. Administrators use a computer console 48 to deploy customizations to individual or groups of services. The computer console 48 presents the administrator with all available customizations by querying the customization repository 42 along a link 50. Using the customization repository 42 in this way decouples development and deployment of customizations. When a new customization is created, the developer simply places it in the customization repository 42, which in turn updates the computer console 48 with its availability across link 50. A customization deployment interface 52 has been added to every service. In one embodiment, the customization deployment interface 52 is implemented within the middleware libraries so that no changes to the application 10 are required. The customization deployment interface is essentially equivalent to the customization control 23 of FIG. 2. The computer console 48 invokes the customization deployment interface 52 through a link 53 to load a customization into the service 54 or into a service's clients 56-1 through 56-n. If the customization is to be loaded into the service 54, the service 54 contacts the customization repository 42 by a link 58 to retrieve the named customization and installs it using the customization control 23 described above for FIG. 2. If the customization is for the service's clients 56-1 through 56-n, the service 54 propagates this request to its clients 56-1 through 56-n via a link 60 to each client. When each client receives the request, it behaves

like the service 54 by retrieving the customization from the customization repository 42 through a link 62 and installing it via the customization control 23.

Downloading executable code into the service 54 or the client 56 requires the service 54 to be able to trust that code not to be malicious. It is assumed that the customization repository 42 is a trusted entity, so code retrieved from it is safe. An alternative, and more scalable, architecture includes the executable code with the customization command to the service 54. Maintaining the trust relationship, when arbitrary entities can request customization operations, will be more difficult than with the well-known repository so additional security could be established by digitally signing the customization modules.

A first example of how customization is used would be what is referred to as "caching", which would be for performance purposes. When a request comes in, the customization module 26-n can look at the set of input data to the request and can determine if there has been the exact same request that has the exact same input data, in the past. The customization module 26-n can return the same result that it saw last time if it can make assumptions about the result not changing because the inputs do not change. If it has not seen this set of input data in the past, it goes ahead and pushes the request into the sub-system 12. Then when the customization module 26-n receives the result, it remembers the result that it received so that on a future invocation when it sees the same inputs it can simply return the value that it saw in this invocation.

A second example of how customization is used is for availability purposes or high availability purposes. An individual customization module 26-4 can be written that will mask or make failures of an underlying sub-system not visible to higher layers of the system such as the application or indeed the actual user of the application.

The customization module 26-4 will receive the call from the application 10 before it goes to the sub-system 12. In a normal case it will simply forward that invocation on to the sub-system 12. But in the case of a failure where the sub-system 12 does not correctly handle the invocation, its status returned from the operation as a failure or a failure to return within a specified time. The customization module 26-4 can detect that situation, detect and find a new sub-system 12-1 through 12-k, whether dynamically at run-time by going out and asking another service (not shown) where a replacement is, or statically from a static list of different sub-systems that are all equally able to handle a request. In any case, it will have the knowledge of an alternative sub-system and undetected it will take the original invocation request and make that request to a second sub-system and with the expectation that one will

perform correctly and then the result can be returned back up to the application 10. So by intercepting or seeing these failures in the customization module, a new sub-system can be found to actually run the operation, and return the result without the application 10 actually knowing that a failure ever occurred.

Failures of hardware and software components are inevitable, so the goal of high availability systems is to mask or hide failures as often as possible so that they do not interfere with the tasks users wish to perform. In a distributed system the goal is for remote services to appear to be operating continually even when the reality is that components that make up the service are not functioning. A crucial part of achieving this is detecting when a failure occurs, discovering an alternate server, and continuing operations with this new server. This process is referred to as "failover". Assuming that the servers are stateless between client invocations, and that they have no side-effects such as updates to a database or file system, a collection of servers and a basic failover may be all that is needed for a service to be continuously available.

As a concrete example of the invention in practice, consider an automated teller machine. It must communicate with computer systems at a bank to complete an individual's transaction. So, in the context of the invention, the teller machine is the application 10 with the bank's computers acting as the sub-system 12 that executes a transaction. By use of the invention, the teller machines can be made more reliable by use of a fail-over mechanism as described above. This fail-over mechanism must be somewhat more sophisticated than described previously to guarantee that no transactions are lost or duplicated, but would be obvious to one having ordinary skill in the art based on the description herein. Teller machines can be instructed via a customization console 48 to install a fail-over module as described. This module can be written to use alternate bank computers (for example, at different geographic locations) to process a transaction if the first attempt fails. By using the invention, these customization modules can be changed after the teller machine is in place to change the fail-over policy such as using entirely new set of back-up locations in case of failures. Because this is permitted to be done on-line, and remotely, these changes can be made without taking the teller machines out of service and without forcing people to physically visit and change the machines providing tremendous time and cost savings, and leading to teller machines that are functioning more often.

The computer system 40 is described as a system for customizing on-line computer services because the disclosed invention permits computer systems to be customized during

operation without interrupting the service that they are providing. Further, those who create customizations, the customization developers, and who populate the customization repository 42 with customizations can work on the computer system 40 at the same time that customizations are being produced and tested.

- 5 While the invention has been described in conjunction with a specific best mode, it is to be understood that many alternatives, modifications, and variations will be apparent to those skilled in the art in light of the foregoing description. Accordingly, it is intended to embrace all such alternatives, modifications, and variations which fall within the spirit and scope of the included claims. All matters set forth herein or shown in the accompanying
- 10 drawings are to be interpreted in an illustrative and non-limiting sense.

11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100
101
102
103
104
105
106
107
108
109
110
111
112
113
114
115
116
117
118
119
120
121
122
123
124
125
126
127
128
129
130
131
132
133
134
135
136
137
138
139
140
141
142
143
144
145
146
147
148
149
150
151
152
153
154
155
156
157
158
159
160
161
162
163
164
165
166
167
168
169
170
171
172
173
174
175
176
177
178
179
180
181
182
183
184
185
186
187
188
189
190
191
192
193
194
195
196
197
198
199
200
201
202
203
204
205
206
207
208
209
210
211
212
213
214
215
216
217
218
219
220
221
222
223
224
225
226
227
228
229
230
231
232
233
234
235
236
237
238
239
240
241
242
243
244
245
246
247
248
249
250
251
252
253
254
255
256
257
258
259
260
261
262
263
264
265
266
267
268
269
270
271
272
273
274
275
276
277
278
279
280
281
282
283
284
285
286
287
288
289
290
291
292
293
294
295
296
297
298
299
300
301
302
303
304
305
306
307
308
309
310
311
312
313
314
315
316
317
318
319
320
321
322
323
324
325
326
327
328
329
330
331
332
333
334
335
336
337
338
339
340
341
342
343
344
345
346
347
348
349
350
351
352
353
354
355
356
357
358
359
360
361
362
363
364
365
366
367
368
369
370
371
372
373
374
375
376
377
378
379
380
381
382
383
384
385
386
387
388
389
390
391
392
393
394
395
396
397
398
399
400
401
402
403
404
405
406
407
408
409
410
411
412
413
414
415
416
417
418
419
420
421
422
423
424
425
426
427
428
429
430
431
432
433
434
435
436
437
438
439
440
441
442
443
444
445
446
447
448
449
450
451
452
453
454
455
456
457
458
459
460
461
462
463
464
465
466
467
468
469
470
471
472
473
474
475
476
477
478
479
480
481
482
483
484
485
486
487
488
489
490
491
492
493
494
495
496
497
498
499
500
501
502
503
504
505
506
507
508
509
510
511
512
513
514
515
516
517
518
519
520
521
522
523
524
525
526
527
528
529
530
531
532
533
534
535
536
537
538
539
540
541
542
543
544
545
546
547
548
549
550
551
552
553
554
555
556
557
558
559
560
561
562
563
564
565
566
567
568
569
570
571
572
573
574
575
576
577
578
579
580
581
582
583
584
585
586
587
588
589
590
591
592
593
594
595
596
597
598
599
600
601
602
603
604
605
606
607
608
609
610
611
612
613
614
615
616
617
618
619
620
621
622
623
624
625
626
627
628
629
630
631
632
633
634
635
636
637
638
639
640
641
642
643
644
645
646
647
648
649
650
651
652
653
654
655
656
657
658
659
660
661
662
663
664
665
666
667
668
669
670
671
672
673
674
675
676
677
678
679
680
681
682
683
684
685
686
687
688
689
690
691
692
693
694
695
696
697
698
699
700
701
702
703
704
705
706
707
708
709
710
711
712
713
714
715
716
717
718
719
720
721
722
723
724
725
726
727
728
729
730
731
732
733
734
735
736
737
738
739
740
741
742
743
744
745
746
747
748
749
750
751
752
753
754
755
756
757
758
759
760
761
762
763
764
765
766
767
768
769
770
771
772
773
774
775
776
777
778
779
780
781
782
783
784
785
786
787
788
789
790
791
792
793
794
795
796
797
798
799
800
801
802
803
804
805
806
807
808
809
810
811
812
813
814
815
816
817
818
819
820
821
822
823
824
825
826
827
828
829
830
831
832
833
834
835
836
837
838
839
840
841
842
843
844
845
846
847
848
849
850
851
852
853
854
855
856
857
858
859
860
861
862
863
864
865
866
867
868
869
870
871
872
873
874
875
876
877
878
879
880
881
882
883
884
885
886
887
888
889
890
891
892
893
894
895
896
897
898
899
900
901
902
903
904
905
906
907
908
909
910
911
912
913
914
915
916
917
918
919
920
921
922
923
924
925
926
927
928
929
930
931
932
933
934
935
936
937
938
939
940
941
942
943
944
945
946
947
948
949
950
951
952
953
954
955
956
957
958
959
960
961
962
963
964
965
966
967
968
969
970
971
972
973
974
975
976
977
978
979
980
981
982
983
984
985
986
987
988
989
990
991
992
993
994
995
996
997
998
999
1000
1001
1002
1003
1004
1005
1006
1007
1008
1009
1010
1011
1012
1013
1014
1015
1016
1017
1018
1019
1020
1021
1022
1023
1024
1025
1026
1027
1028
1029
1030
1031
1032
1033
1034
1035
1036
1037
1038
1039
1040
1041
1042
1043
1044
1045
1046
1047
1048
1049
1050
1051
1052
1053
1054
1055
1056
1057
1058
1059
1060
1061
1062
1063
1064
1065
1066
1067
1068
1069
1070
1071
1072
1073
1074
1075
1076
1077
1078
1079
1080
1081
1082
1083
1084
1085
1086
1087
1088
1089
1090
1091
1092
1093
1094
1095
1096
1097
1098
1099
1100
1101
1102
1103
1104
1105
1106
1107
1108
1109
1110
1111
1112
1113
1114
1115
1116
1117
1118
1119
1120
1121
1122
1123
1124
1125
1126
1127
1128
1129
1130
1131
1132
1133
1134
1135
1136
1137
1138
1139
1140
1141
1142
1143
1144
1145
1146
1147
1148
1149
1150
1151
1152
1153
1154
1155
1156
1157
1158
1159
1160
1161
1162
1163
1164
1165
1166
1167
1168
1169
1170
1171
1172
1173
1174
1175
1176
1177
1178
1179
1180
1181
1182
1183
1184
1185
1186
1187
1188
1189
1190
1191
1192
1193
1194
1195
1196
1197
1198
1199
1200
1201
1202
1203
1204
1205
1206
1207
1208
1209
1210
1211
1212
1213
1214
1215
1216
1217
1218
1219
1220
1221
1222
1223
1224
1225
1226
1227
1228
1229
1230
1231
1232
1233
1234
1235
1236
1237
1238
1239
1240
1241
1242
1243
1244
1245
1246
1247
1248
1249
1250
1251
1252
1253
1254
1255
1256
1257
1258
1259
1260
1261
1262
1263
1264
1265
1266
1267
1268
1269
1270
1271
1272
1273
1274
1275
1276
1277
1278
1279
1280
1281
1282
1283
1284
1285
1286
1287
1288
1289
1290
1291
1292
1293
1294
1295
1296
1297
1298
1299
1300
1301
1302
1303
1304
1305
1306
1307
1308
1309
1310
1311
1312
1313
1314
1315
1316
1317
1318
1319
1320
1321
1322
1323
1324
1325
1326
1327
1328
1329
1330
1331
1332
1333
1334
1335
1336
1337
1338
1339
1340
1341
1342
1343
1344
1345
1346
1347
1348
1349
1350
1351
1352
1353
1354
1355
1356
1357
1358
1359
1360
1361
1362
1363
1364
1365
1366
1367
1368
1369
1370
1371
1372
1373
1374
1375
1376
1377
1378
1379
1380
1381
1382
1383
1384
1385
1386
1387
1388
1389
1390
1391
1392
1393
1394
1395
1396
1397
1398
1399
1400
1401
1402
1403
1404
1405
1406
1407
1408
1409
1410
1411
1412
1413
1414
1415
1416
1417
1418
1419
1420
1421
1422
1423
1424
1425
1426
1427
1428
1429
1430
1431
1432
1433
1434
1435
1436
1437
1438
1439
1440
1441
1442
1443
1444
1445
1446
1447
1448
1449
1450
1451
1452
1453
1454
1455
1456
1457
1458
1459
1460
1461
1462
1463
1464
1465
1466
1467
1468
1469
1470
1471
1472
1473
1474
1475
1476
1477
1478
1479
1480
1481
1482
1483
1484
1485
1486
1487
1488
1489
1490
1491
1492
1493
1494
1495
1496
1497
1498
1499
1500
1501
1502
1503
1504
1505
1506
1507
1508
1509
1510
1511
1512
1513
1514
1515
1516
1517
1518
1519
1520
1521
1522
1523
1524
1525
1526
1527
1528
1529
1530
1531
1532
1533
1534
1535
1536
1537
1538
1539
1540
1541
1542
1543
1544
1545
1546
1547
1548
1549
1550
1551
1552
1553
1554
1555
1556
1557
1558
1559
1560
1561
1562
1563
1564
1565
1566
1567
1568
1569
1570
1571
1572
1573
1574
1575
1576
1577
1578
1579
1580
1581
1582
1583
1584
1585
1586
1587
1588
1589
1590
1591
1592
1593
1594
1595
1596
1597
1598
1599
1600
1601
1602
1603
1604
1605
1606
1607
1608
1609
1610
1611
1612
1613
1614
1615
1616
1617
1618
1619
1620
1621
1622
1623
1624
1625
1626
1627
1628
1629
1630
1631
1632
1633
1634
1635
1636
1637
1638
1639
1640
1641
1642
1643
1644
1645
1646
1647
1648
1649
1650
1651
1652
1653
1654
1655
1656
1657
1658
1659
1660
1661
1662
1663
1664
1665
1666
1667
1668
1669
1670
1671
1672
1673
1674
1675
1676
1677
1678
1679
1680
1681
1682
1683
1684
1685
1686
1687
1688
1689
1690
1691
1692
1693
1694
1695
1696
1697
1698
1699
1700
1701
1702
1703
1704
1705
1706
1707
1708
1709
1710
1711
1712
1713
1714
1715
1716
1717
1718
1719
1720
1721
1722
1723
1724
1725
1726
1727
1728
1729
1730
1731
1732
1733
1734
1735
1736
1737
1738
1739
1740
1741
1742
1743
1744
1745
1746
1747
1748
1749
1750
1751
1752
1753
1754
1755
1756
1757
1758
1759
1760
1761
1762
1763
1764
1765
1766
1767
1768
1769
1770
1771
1772
1773
1774
1775
1776
1777
1778
1779
1780
1781
1782
1783
1784
1785
1786
1787
1788
1789
1790
1791
1792
1793
1794
1795
1796
1797
1798
1799
1800
1801
1802
1803
1804
1805
1806
1807
1808
1809
1810
1811
1812
1813
1814
1815
1816
1817
1818
1819
1820
1821
1822
1823
1824
1825
1826
1827
1828
1829
1830
1831
1832
1833
1834
1835
1836
1837
1838
1839
1840
1841
1842
1843
1844
1845
1846
1847
1848
1849
1850
1851
1852
1853
1854
1855
1856
1857
1858
1859
1860
1861
1862
1863
1864
1865
1866
1867
1868
1869
1870
1871
1872
1873
1874
1875
1876
1877
1878
1879
1880
1881
1882
1883
1884
1885
1886
1887
1888
1889
1890
1891
1892
1893
1894
1895
1896
1897
1898
1899
1900
1901
1902
1903
1904
1905
1906
1907
1908
1909
1910
1911
1912
1913
1914
1915
1916
1917
1918
1919
1920
1921
1922
1923
1924
1925
1926
1927
1928
1929
1930
1931
1932
1933
1934
1935
1936
1937
1938
1939
1940
1941
1942
1943
1944
1945
1946
1947
1948
1949
1950
1951
1952
1953
1954
1955
1956
1957
1958
1959
1960
1961
1962
1963
1964
1965
1966
1967
1968
1969
1970
1971
1972
1973
1974
1975
1976
1977
1978
1979
1980
1981
1982
1983
1984
1985
1986
1987
1988
1989
1990
1991
1992
1993
1994
1995
1996
1997
1998
1999
2000
2001
2002
2003
2004
2005
2006
2007
2008
2009
2010
2011
2012
2013
2014
2015
2016
2017
2018
2019
2020
2021
2022
2023
2024
2025
2026
2027
2028
2029
2030
2031
2032
2033
2034
2035
2036
2037
2038
2039
2040
2041
2042
2043
2044
2045
2046
2047
2048
2049
2050
2051
2052
2053
2054
2055
2056
2057
2058
2059
2060
2061
2062
2063
2064
2065
2066
2067
2068
2069
2070
2071
2072
2073
2074
2075
2076
2077
2078
2079
2080
2081
2082
2083
2084
2085
2086
2087
2088
2089
2090
2091
2092
2093
2094
2095
2096
2097
2098
2099
2100
2101
2102
2103
2104
2105
2106
2107
2108
2109
2110
2111
2112
2113
2114
2115
2116
2117
2118
2119
2120
2121
2122
2123
2124
2125
2126
2127
2128
2129
2130
2131
2132
2133
2134
2135
2136
2137
2138
2139
2140
2141
2142
2143
2144
2145
2146
2147
2148
2149
2150
2151
2152
2153
2154
2155
2156
2157
2158
2159
2160
2161
2162
2163
2164
2165
2166
2167
2168
2169
2170
2171
2172
2173
2174
2175
2176
2177
2178
2179
2180
2181
2182
2183
2184
2185
2186
2187
2188
2189
2190
2191
2192
2193
2194
2195
2196
2197
2198
2199
2200
2201
2202
2203
2204
2